UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/029,497	12/21/2001	Sivaram Krishnan	36992.00134	4638

30256 7590 01/18/2007

SQUIRE, SANDERS & DEMPSEY L.L.P
PATENT DEPARTMENT
ONE MARITIME PLAZA, SUITE 300
SAN FRANCISCO, CA 94111-3492

EXAMINER
GUILL, RUSSELL L

ART UNIT	PAPER NUMBER
2123	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	01/18/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/029,497	KRISHNAN, SIVARAM	
	Examiner	Art Unit	
	Russ Guill	2123	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *13 November 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3-5,7-9,24,25,27-32,36 and 37* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3-5,7-9,24,25,27-32,36 and 37* is/are rejected.

7)☒ Claim(s) *4,30* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *21 December 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to an Amendment filed November 13, 2006. Claims 2, 10, 26 and 33 - 35 were canceled. Claims 36 – 37 were added. Claims 1, 3 – 5, 7 – 9, 24 – 25, 27 – 32 and 36 - 37 are pending. Claims 1, 3 – 5, 7 – 9, 24 – 25, 27 – 32 and 36 - 37 have been examined. Claims 1, 3 – 5, 7 – 9, 24 – 25, 27 – 32 and 36 - 37 have been rejected. Claims 4 and 32 are allowable over the prior art of record.

2. This Office Action is NON-final as a result of amended rejections under 35 U.S.C. § 103.

3. **The Examiner would like to thank the Applicant for the well-presented amendment, which was useful in the examination process. The Examiner appreciates the effort to carefully analyze the Office Action, and make appropriate arguments and amendments. The Examiner also thanks the Applicant for the time taken to perform an interview, which is useful to expedite the examination process.**

Response to Remarks

4. Regarding claims 1 and 24 rejected under 35 USC § 101:

4.1. Applicant's arguments on page 8 regarding independent claims 1 and 24 have been fully reviewed, and are persuasive. However, upon further consideration, new rejections are made as described below. Also, regarding Applicant's arguments on page 9, the Examiner respectfully disagrees with portions, as follows. Briefly, Altman appears to teach hardware emulation (page 40, left-side column, last sentence in the bullet item). Further, art is provided with this Office Action to support the equivalence of hardware and software.

Claim Objections

5. Claim 30 is objected to for the following minor informalities: Claim 30 depends from a canceled claim 26. For the purpose of claim examination, the claim is interpreted as depending from claim 24.

Claim Rejections - 35 USC § 101

6. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

7. Claims 1, 3 – 5, 7 – 9, 24 – 25, 27 – 32 and 36 - 37 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

7.1. Regarding independent claims 1 and its dependent claims, the recited method appears to contain abstract ideas such as producing first dynamic execution information. Therefore, to be statutory, the claims must be directed to a practical application producing a concrete, useful and tangible result. The claims do not appear to produce a tangible result needed to support a practical application. Changing the hardware emulation does not appear to be a tangible result because a process is not tangible.

7.2. Regarding independent claims 24 and its dependent claims, the recited computer system appears to contain abstract ideas such as producing first dynamic execution information. Therefore, to be statutory, the claims must be directed to a practical application producing a

concrete, useful and tangible result. The claims do not appear to produce a tangible result needed to support a practical application. Changing the hardware emulation code generator does not appear to produce tangible result because the change appears to be simply changing bit settings, which is not tangible.

7.3. Regarding independent claim 32 and its dependent claims, the recited method appears to contain abstract ideas such as producing first dynamic execution information. Therefore, to be statutory, the claim must be directed to a practical application producing a concrete, useful and tangible result. The claim does not appear to produce a tangible result needed to support a practical application. Broadly interpreted, a computer system can be entirely software, and changing software would not provide a tangible result. A claim that can be interpreted to include both statutory and non-statutory embodiments must be amended to only include statutory interpretations.

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. **Claims 1 and 24** are rejected under 35 U.S.C. 103(a) as being unpatentable over Altman (Altman, Erik R.; Kaeli, David; Sheffer, Yaron; "Welcome to the opportunities of Binary Translation", March 2000,

IEEE Computer), in view of Mattson (U.S. Patent Number 6,115,809), further in view of admitted prior art of the Applicant.

9.1. Regarding **claim 1**:

9.2. Altman appears to teach:

9.2.1. a method performed by a computer system having a hardware emulation *(page 40, left-side column, last line, "provide a special processor mode to execute legacy code on the new processor")*.

9.2.2. Obtaining a first set of one or more emulated instructions derived from an original set of one or more instructions *(page 40 – 41, section labeled "Three Types of Translation")*.

9.2.3. initiating execution of the first set of one or more emulated instructions (*page 41, left-side column, third paragraph, and section labeled Runtime Optimization; it would have been obvious that the emulated sequence of instructions were executed*).

9.2.4. Producing first dynamic execution information in response to executing the first set of emulated instructions *(page 41, section labeled "Profiling", first paragraph)*;

9.2.5. Changing the *software* emulation dynamically for producing a second set of one or more emulated instructions by modifying the first set of one or more emulated instructions in response to said first dynamic execution information (*page 41, section Dynamic Optimization; and page 42, sidebar Native Binary Acceleration*).

9.3. Altman does not specifically teach (missing parts of the limitation are in **bold underline**):

9.3.1. Obtaining a first set of one or more emulated instructions derived from an original set of one or more instructions **using the hardware emulation**;

9.3.2. Changing the **hardware** emulation dynamically for producing a second set of one or more emulated instructions by modifying **at least a parameter of one instruction of** the first set of one or more emulated instructions in response to said first dynamic execution information.

9.4. Admitted prior art of the Applicant appears to teach:

9.4.1. using hardware emulation (*page 3, lines 11 - 20*);

9.5. Mattson appears to teach:

9.5.1. Producing first dynamic execution information in response to executing the first set of instructions (*column 6, lines 39 – 43; ". . . techniques may be used to collect branching information . . . during dynamic execution if the code is to be dynamically optimized"*);

9.5.2. Changing the emulation dynamically for producing a second set of one or more emulated instructions by modifying the first set of one or more emulated instructions in response to said first dynamic execution information. (*column 9, lines 49 – 65, especially lines 60 - 65; column 9, lines 23 – 33; please note that Just-in-Time compiling Java bytecodes to run on a target machine is a form of emulation, as also noted in the Applicant's specification on page 2, lines 15 – 20 and page 3, lines 1 – 3, and also see*

Altman, page 41, sidebar "A Just-in-Time Compiler" regarding compiling Java bytecodes; and also see Burke et al., especially figure 1).

9.5.3. producing a second set of one or more instructions by modifying at least a parameter of one instruction of a first set of one or more instructions in response to first dynamic execution information (*column 2, lines 31 – 42; and column 3, lines 25 – 34; and column 4, lines 1 – 21 [please note that in column 4, lines 1 - 21 the instruction parameter modification is performed by hardware, where the parameter is the prediction]; and column 9, lines 23 – 33*).

9.6. The motivation to use the admitted prior art of the Applicant with the art of Altman would have been the benefit recited in the admitted prior art of the Applicant that emulation through hardware is considerably faster than emulation through software (*page 3, lines 11 - 15*).

9.7. The motivation to use the art of Mattson with the art of Altman would have been the advantages recited in Mattson, including that predictions vary dynamically to changes in the computing environment that may affect branching (*column 3, lines 25 – 31*), and predictions tend to be very accurate when the prediction table is large (*column 3, lines 30 – 33*), which would have been recognized by the ordinary artisan as benefits that result in reduced execution time for programs. Further, the ordinary artisan would have been motivated to use the runtime optimization of Mattson because the ordinary artisan would have understood the benefits of runtime optimization to reduce execution time, thereby saving time and cost.

9.8. To summarize, Mattson appears to teach using dynamic runtime information to perform runtime optimization of emulated instructions (column 9, lines 49 – 65, especially Java). Mattson

further appears to teach using hardware to modify a parameter of an instruction at runtime based upon dynamic execution information (column 4, lines 1 - 21). The admitted prior art of the Applicant teaches using hardware emulation, with the motivation that hardware emulation has the benefit of being considerably faster than software emulation (page 3, lines 10 - 20). These elements appear to provide the essentials of the claimed invention.

9.9. In further support of obviousness, it is important to have the knowledge of the ordinary artisan at the time of invention. The Examiner notes that it was old and well-known by the ordinary artisan at the time of invention to use hardware to modify a parameter of an instruction when optimizing a sequence of instructions at runtime (see art provided previously, David A. Patterson, "Computer Architecture A Quantitative Approach", second edition, 1996, page 232 - 233, section "Name Dependences", especially starting at the sentence that starts with, "This means that instructions . . ."). Further, it was old and well-known by the ordinary artisan at the time of invention that hardware and software are logically equivalent (pages 11 - 12 of Andrew S. Tanenbaum, "Structured Computer Organization", second edition, 1984, Prentice-Hall). Further, it was old and well-known known by the ordinary artisan at the time of invention to collect dynamic runtime information from running code and use the collected information to perform dynamic runtime optimization of the code (page 130, figure 1, Burke et al.; "The Jalapeno Dynamic Optimizing Compiler for Java", 1999, Proceedings of the ACM 1999 Conference on Java Grande; and Arnold et al.; "Adaptive Optimization in the Jalapeno JVM", 2000, Conference on Object Oriented Programming Systems and Languages, pages 47 - 65). Further, it was known to have a hardware emulator perform optimizations similar to Just-in-Time compilers (page 427, Abstract, last sentence; Ramesh Radhakrishnan et al.; "Improving Java Performance Using Hardware Translation", June 2001, Proceedings of the 15th International Conference on Supercomputing, pages 427 - 439).

9.10. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Mattson and the admitted prior art of the Applicant with the art of Altman to produce the claimed invention.

9.11. Regarding **claim 24,** Altman appears to teach:

9.11.1. a processor means for executing a first set of one or more emulated sequence of instructions (***page 41, left-side column, third paragraph, and section labeled Runtime Optimization; it would have been obvious that the emulated sequence of instructions were executed***).

9.11.2. a means for producing dynamic execution information in response to execution of the first set of one or more emulated instructions (***page 41, section labeled "Profiling"***).

9.12. Altman does not specifically teach:

9.12.1. a hardware emulation code generator generating a first set of one or more emulated instructions produced from an original set of one or more instructions;

9.12.2. a means for changing the hardware emulation code generator dynamically to produce a second set of one or more emulated instructions by modifying at least a parameter of one instruction of the first set of one or more emulated instructions in response to said dynamic execution information.

9.13. Admitted prior art of the Applicant appears to teach:

9.13.1. a hardware emulation code generator generating a first set of one or more emulated instructions produced from an original set of one or more instructions (page 3, lines 11 - 20);

9.14. Mattson appears to teach:

9.14.1. a means for producing dynamic execution information in response to execution of a first set of one or more instructions (*column 6, lines 39 – 43; ". . . techniques may be used to collect branching information . . . during dynamic execution if the code is to be dynamically optimized"*);

9.14.2. a means for changing the emulation code generator dynamically to produce a second set of one or more emulated instructions by modifying at least a parameter of one instruction of the first set of one or more emulated instructions in response to said dynamic execution information (*column 9, lines 49 – 65, especially lines 60 - 65; column 9, lines 23 – 33; please note that Just-in-Time compiling Java bytecodes to run on a target machine is a form of emulation, as also noted in the Applicant's specification on page 2, lines 15 – 20 and page 3, lines 1 – 3, and also see Altman, page 41, sidebar "A Just-in-Time Compiler" regarding compiling Java bytecodes*).

9.14.3. producing a second set of one or more instructions by modifying at least a parameter of one instruction of a first set of one or more instructions in response to dynamic execution information (*column 2, lines 31 – 42; and column 3, lines 25 – 34; and column 4, lines 1 – 21 [please note that in column 4, lines 1 - 21 the instruction parameter modification is performed by hardware, where the parameter is the prediction]; and column 9, lines 23 – 33*).

9.15. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Mattson and the admitted prior art of the Applicant with the art of Altman to produce the claimed invention.

10. **Claim 5** is rejected under 35 U.S.C. 103(a) as being unpatentable over Altman as modified by Mattson and admitted prior art of the Applicant as applied to **claims 1 and 24** above, further in view of Kistler (Thomas Kistler et al.; "Continuous Program Optimization: Design and Evaluation", June 2001, IEEE Transactions on Computers).

10.1. Altman as modified by Mattson and admitted prior art of the Applicant teaches a method of producing dynamic execution information in response to executing emulated instructions, and changing the hardware emulation of a computer system dynamically for producing a second set of emulated instructions in response to said first dynamic execution information, as recited in **claims 1 and 24** above.

10.2. The art of Altman is directed toward translating machine instructions for one computer into machine instructions to run on a second computer, including profiling and dynamic optimization ***(pages 40 - 41)***.

10.3. The art of Kistler is directed to dynamic program optimization ***(page 549, Abstract)***.

10.4. Regarding **claim 5**, Altman does not specifically teach:

10.4.1. said steps of executing, producing and changing are conducted recursively on at least some of successive segments of the first set of one or more emulated instructions.

10.5. Regarding **claim 5**, Kistler appears to teach:

10.5.1. steps of executing, producing and changing are conducted recursively on at least some of successive segments of a first set of one or more emulated instructions (*page 549, Abstract, first three sentences*).

10.6. The motivation to use the art of Kistler with the art of Altman as modified by Mattson and admitted prior art of the Applicant would have been the benefits recited in Kistler that the method eliminates some of the most severe performance problems (*page 564, first paragraph*), and can result in real performance improvements (*page 564, third paragraph*).

10.7. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Kistler with the art of Altman as modified by Mattson and admitted prior art of the Applicant to produce the claimed invention.

11. **Claim 7** is rejected under 35 U.S.C. 103(a) as being unpatentable over Altman in view of Mattson and admitted prior art of the Applicant as applied to **claims 1 and 24** above, further in view of Smith (U.S. Patent Number 4,370,711).

11.1. Altman as modified by Mattson and admitted prior art of the Applicant teaches a method of producing dynamic execution information in response to executing emulated instructions, and changing the hardware emulation of a computer system dynamically for producing a second set of emulated instructions in response to said first dynamic execution information, as recited in **claims 1 and 24** above.

11.2. The art of Altman is directed toward translating machine instructions for one computer into machine instructions to run on a second computer, including profiling and dynamic optimization ***(pages 40 - 41)***.

11.3. The art of Smith is directed to branch prediction ***(page 1, Abstract)***.

11.4. Regarding **claim 7**, Altman does not specifically teach:

11.4.1. said step of producing, produces branch prediction information;

11.4.2. said step of changing, changes condition codes of the branch prediction information.

11.5. Regarding **claim 7**, Smith appears to teach:

11.5.1. producing branch prediction information (***column 1, lines 44 - 67, and column 2, lines 1 - 2***);

11.5.2. changing condition codes of the branch prediction information (***column 1, lines 44 - 67, and column 2, lines 1 - 2; the branch prediction information is a form of condition code***);

11.6. The motivation to use the art of Smith with the art of Altman as modified by Mattson and admitted prior art of the Applicant would have been the benefits recited in Smith including improved branch prediction mechanism with a high prediction accuracy to minimize the time loss caused by incorrect predictions (**column 1, lines 17 -21**), and using less and simpler hardware than another technique might (**column 3, lines 26 - 29**), which would have been recognized as benefit by the ordinary artisan to reduce code execution time.

11.7. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Smith with the art of Altman as modified by Mattson and admitted prior art of the Applicant to produce the claimed invention.

12. **Claim 3, 8 - 9, 27 - 30 and 36 - 37** is rejected under 35 U.S.C. 103(a) as being unpatentable over Altman as modified by Mattson and admitted prior art of the Applicant as applied to **claims 1 and 24** above, further in view of Patterson (David A. Patterson et al.; "Computer architecture A Quantitative Approach", second edition, 1996, Morgan Kaufmann Publishers).

12.1. Altman as modified by Mattson and admitted prior art of the Applicant teaches a method of producing dynamic execution information in response to executing emulated instructions, and changing the hardware emulation of a computer system dynamically for producing a second set of emulated instructions in response to said first dynamic execution information, as recited in **claims 1 and 24** above.

12.2. The art of Altman is directed toward translating machine instructions for one computer into machine instructions to run on a second computer, including profiling and dynamic optimization *(pages 40 - 41)*.

12.3. The art of Patterson is directed to common knowledge in the art of computer architecture *(title)*.

12.4. In addition to the motivations provided below, the ordinary artisan would have been motivated to use the art of Patterson with the art of Altman as modified by Mattson and admitted prior art of the Applicant as applied to **claims 1 - 3, 10, 24 and 26 - 27** above because the ordinary artisan would have used any and all known code optimization methods in the

runtime optimization provided by the art of Altman as modified by Mattson and admitted prior art of the Applicant in order to reduce code execution time. The ordinary artisan would have been motivated to search the literature for known code optimization methods.

12.5. Regarding **claims 3 and 27**, Altman does not specifically teach:

12.5.1. modifying at least a register field of one instruction of the first set of one or more emulated instructions.

12.6. Patterson appears to teach:

12.6.1. modifying at least a register field of one instruction of the first set of one or more emulated instructions (*pages 232 – 233, section "Name Dependences"*).

12.7. The motivation to use the art of Patterson with the art of Altman as modified by Mattson would have been the benefit recited in Patterson of instructions involved in a name dependence can execute simultaneously if the register number used in the instructions is changed so the instructions do not conflict (*page 232, section "Name Dependences"*), which would have been recognized as a valuable benefit by the ordinary artisan to reduce code execution time.

12.8. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Patterson with the art of Altman as modified by Mattson and admitted prior art of the Applicant to produce the claimed invention.

12.9. Regarding **claim 9**, Altman does not specifically teach:

12.9.1. said step of producing, produces a history of branch prediction dynamic execution information;

12.9.2. said step of changing, generates a branch prediction likelihood code for a group of branches that may be different from any branch prediction of the members of the group.

12.10. Patterson appears to teach:

12.10.1. producing a history of branch prediction dynamic execution information (*page 262, section Basic Branch Prediction and Branch-Prediction Buffers*);

12.10.2. generating a branch prediction likelihood code for a group of branches that may be different from any branch prediction of the members of the group (**pages 266 – 268**).

12.11. The motivation to use the art of Patterson with the art of Altman as modified by Mattson would have been the benefit recited in Patterson of reducing branch penalties (*page 262, section 4.3, title*), which would have been recognized as a valuable benefit by the ordinary artisan to reduce code execution time.

12.12. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Patterson with the art of Altman as modified by Mattson and admitted prior art of the Applicant to produce the claimed invention.

12.13. Regarding **claim 8**, Altman does not specifically teach:

12.13.1. the step of producing produces a history of register allocation information.

12.13.2. the step of changing changes register allocation.

12.14. Regarding **claim 8**, Patterson appears to teach:

12.14.1. producing a history of register allocation information *(page 233, top half of page)*.

12.14.2. changing register allocation *(page 233, bottom half of page, and page 232, paragraph that starts with "Both antidependences . . ." recites that the process may be performed dynamically by hardware)*.

12.15. The motivation to use the art of Patterson with the art of Altman would have been the benefit recited in Patterson that resolving dependences allows simultaneous execution of instructions (*page 232, paragraph that starts with "Both antidependences . . ."; and page 229, section Dependences, first paragraph)*, which would have been recognized as a valuable benefit by the ordinary artisan.

12.16. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Patterson with the art of Altman as modified by Mattson and admitted prior art of the Applicant to produce the claimed invention.

12.17. Regarding **claim 28**, Altman does not specifically teach:

12.17.1. cycling allocation of registers in a pool of registers as some of successively identified registers in the first set of one or more emulated instructions.

12.18. Regarding **claim 28**, Patterson appears to teach:

12.18.1. cycling allocation of registers in a pool of registers as some of successively identified registers in the emulated sequence of instructions *(page 233, bottom half of page)*.

12.19. Regarding **claim 29**, Altman does not specifically teach:

12.19.1. producing a history of temporary register allocation information.

12.19.2. changes a register parameter of one instruction of the first set of one or more emulated instructions.

12.20. Regarding **claim 29**, Patterson appears to teach:

12.20.1. producing a history of register allocation information *(page 233, top half of page)*.

12.20.2. changing a register parameter of one instruction of a first set of one or more instructions *(page 233, bottom half of page, and page 232, paragraph that starts with "Both antidependences . . ." recites that the process may be performed dynamically by hardware)*.

12.21. Regarding **claim 30**, Altman appears to teach:

12.21.1. an emulation code generator for generating the first set of one or more instructions that is executable with a first instruction set from the set of one or more original instructions that is executable with a different instruction set *(page 40, title; and abstract directly beneath the title; and left-side column, paragraph 3, definition of "binary translation")*.

12.21.2. modifying the first set of one or more emulated instructions in response to at least the historical register usage information *(page 41, section "Dynamic optimization")*.

12.21.3. execution of first set of one or more emulated instructions (*page 41, left-side column, third paragraph, and section labeled Runtime Optimization; it would have been obvious that the emulated sequence of instructions were executed*).

12.22. Regarding **claim 30**, Altman does not specifically teach:

12.22.1. generating historical register usage information regarding register status.

12.23. Regarding **claim 30**, Patterson appears to teach:

12.23.1. generating historical register usage information regarding register status during execution (*page 233, bottom half of page, and page 232, paragraph that starts with "Both antidependences . . ." recites that the process may be performed dynamically by hardware*).

12.24. Regarding **claim 36**:

12.25. Altman does not specifically teach:

12.25.1. modifying a plurality of parameters of some instructions of the first set of one or more emulated instructions.

12.26. Patterson appears to teach:

12.26.1. modifying a plurality of parameters of some instructions of the first set of one or more emulated instructions (*pages 232 – 233, section "Name Dependences"*, **especially the example on page 233 which demonstrates the limitation**).

12.27. Regarding **claim 37**:

12.28. Altman does not specifically teach:

12.28.1. means for changing the hardware emulation code generator dynamically to produce a second set of one or more emulated instructions by modifying a plurality of parameters of some instructions of the first set of one or more emulated instructions.

12.29. Patterson appears to teach:

12.29.1. means for changing the code generator dynamically to produce a second set of one or more instructions by modifying a plurality of parameters of some instructions of the first set of one or more instructions (*pages 232 – 233, section "Name Dependences"*, especially the example on page 233 which demonstrates the limitation).

13. **Claims 25 and 31** are rejected under 35 U.S.C. 103(a) as being unpatentable over Altman as modified by Mattson and admitted prior art of the Applicant, as applied to **claims 1 and 24** above, further in view of Conte (Conte, Thomas M.; Patel, Burzin A.; Cox, J. Stan; "Using Branch Handling Hardware to Support Profile-Driven Optimization", 1994, Proceedings of the 1994 27th annual international symposium on microarchitecture).

13.1. Altman as modified by Mattson and admitted prior art of the Applicant teaches a method of producing dynamic execution information in response to executing emulated instructions, and changing the hardware emulation of a computer system dynamically for producing a second set

of emulated instructions in response to said first dynamic execution information, as recited in **claims 1 and 24** above.

13.2. The art of Altman is directed toward translating machine instructions for one computer into machine instructions to run on a second computer, including profiling and dynamic optimization ***(pages 40 - 41)***.

13.3. The art of Conte is directed to using branch handling hardware to support profile-driven optimization ***(page 1, Title)***.

13.4. Regarding **claim 25**, Altman does not specifically teach:

13.4.1. maintaining a record of branch addresses in the first set of one or more emulated instructions historically correlated to whether branches were taken during execution of the first set of one or more emulated instructions.

13.4.2. means for changing a likelihood condition code of the branch prediction information for at least one of the branches.

13.5. Regarding **claim 25**, Conte appears to teach:

13.5.1. means for maintaining a record of branch addresses in a first set of one or more emulated instructions historically correlated to whether branches were taken during execution of the first set of one or more emulated instructions ***(page 2, section 2 "Branch Prediction and Profiling"; and figure 1; and figure 2)***.

13.5.2. means for changing a likelihood condition code of the branch prediction information for at least one of the branches ***(page 2, section 2 "Branch Prediction and Profiling"; and figure 1; and figure 2)***.

13.6. The art of Conte and the art of Altman are analogous art because they both contain the problem of profiling and optimization.

13.7. The motivation to use the art of Conte with the art of Altman would have been the benefit recited in Conte that using the specified branch prediction method produces a dramatic effect in achieving 96% accuracy in branch prediction (*page 2, section 2.1 Contemporary branch handling mechanisms, second paragraph*) which would have been recognized as a benefit by the ordinary artisan to reduce code execution time.

13.8. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Conte with the art of Altman as modified by Mattson and admitted prior art of the Applicant to produce the claimed invention.

13.9. Regarding **claim 31**, Altman appears to teach:

13.9.1. an emulation code generator for generating the first set of one or more emulated instructions that is executable with a first instruction set from the set of one or more original instructions that is executable with a different second instruction set (*page 40, title; and abstract directly beneath the title; and left-side column, paragraph 3, definition of "binary translation"*).

13.10. Regarding **claim 31**, Altman does not specifically teach:

13.10.1. generating historical branch prediction dynamic execution information regarding likelihood of branches taken during execution of the first set of one or more emulation instructions.

13.10.2. generating a branch prediction likelihood code for a group of branches that may be different from any branch prediction of the members of the group and is dependent upon a combined effect of the branch predictions of the members of the group.

13.11. Regarding **claim 31**, Conte appears to teach:

13.11.1. generating historical branch prediction dynamic execution information regarding likelihood of branches taken during execution of the first set of one or more emulation instructions *(pages 3 – 4, section 3 Using Branch Prediction Hardware for Profiling; and page 2, section 2 Branch Prediction and Profiling)*.

13.11.2. generating a branch prediction likelihood code for a group of branches that may be different from any branch prediction of the members of the group and is dependent upon a combined effect of the branch predictions of the members of the group *(page 2, section 2.1 Contemporary branch handling mechanisms, second paragraph; and figure 2)*.

13.11.2.1. Regarding *(page 2, section 2.1 Contemporary branch handling mechanisms, second paragraph; and figure 2)*; it would have been obvious to generate a branch prediction likelihood code for a group of branches that may be different from any branch prediction of the members of the group and is dependent upon a combined effect of the branch predictions of the members of the group.

Allowable Subject Matter

14. Claims 4 and 32 are allowable over the prior art of record.

15. As allowable subject matter has been indicated, applicant's reply must either comply with all formal requirements or specifically traverse each requirement not complied with. See 37 CFR 1.111(b) and MPEP § 707.07(a).

16. Claim 4 is objected to as being dependent upon a rejected base claim, but would be allowable over the prior art of record if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

17. An Examiner's statement of reasons for indicating allowable subject matter was given in the previous Office Action dated July 12, 2006.

Conclusion

18. The prior art made of record and not relied upon is considered pertinent to the applicant's disclosure:

18.1. Andrew S. Tanenbaum, "Structured Computer Organization", second edition, 1984, Prentice-Hall, pages 10 - 12; teaches that hardware and software are logically equivalent.

18.2. Burke et al.; "The Jalapeno Dynamic Optimizing Compiler for Java", 1999, Proceedings of the ACM 1999 Conference on Java Grande, pages 129 - 141; teaches collecting dynamic runtime information on running code, and using the dynamic runtime information to dynamically optimize the code during runtime.

18.3. Arnold et al.; "Adaptive Optimization in the Jalapeno JVM", 2000, Conference on Object Oriented Programming Systems and Languages, pages 47 - 65; teaches collecting dynamic runtime information on running code, and using the dynamic runtime information to dynamically optimize the code during runtime.

18.4. Ramesh Radhakrishnan et al.; "Improving Java Performance Using Hardware Translation", June 2001, Proceedings of the 15th International Conference on Supercomputing, pages 427 - 439; teaches a hardware emulator that optimizes code.

19. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Russ Guill whose telephone number is 571-272-7955. The examiner can normally be reached on Monday - Friday 10:00 AM - 6:30 PM.

20. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Paul Rodriguez can be reached on 571-272-3753. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300. Any inquiry of a general nature or relating to the status of this application should be directed to the TC2100 Group Receptionist: 571-272-2100.

21. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

RG

ZOILA CABRERA
PRIMARY EXAMINER
TECHNOLOGY CENTER 2100

01/12/07

Russ Guill
Examiner
Art Unit 2123